Final Project

NYU's Intro to Computer Science - Fall 2022
Section 6 & 9

Design and implement an OOP video graphic game using Processing!

Important deadlines

Only one submission from one team is needed- team leader should submit the files needed by the due date!
Please note that all deadlines are strict and there will be no extensions!

Make sure to upload a .zip folder with all the required items for the complete game by each deadline (by midnight) on Bright space!

- **Project proposal due via Bright Space: 12/7th**, by midnight (No extensions)!

- **Weekly report** due by **12/12** by midnight (a pdf file with all the finished code of the game. Make sure to have the design done, two classes and a simple play working of your game).

- **Project Final Submission due via Bright Space on 12/17th, by midnight:**
  Make sure to zip all the files (The complete game with all files needed for this assignment. Include the UML for all classes as a .pdf, responsibilities for each team members (.pdf), and include also any extra credit such as added classes above 3 class (3 classes is the minimum required classes for the project), if you included sound files or if you created your own images and so on. Make sure that all images and sound files are small files, otherwise it will slow the game. The zip folder should be submitted to bright space by the due date.

Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment.

Any document and program code that you submit must be fully written by yourself. You can, of course, discuss your work with fellow students, as long as these discussions are restricted to general solution techniques. Put differently, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution. That means, you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form.
Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU (see https://cs.nyu.edu/home/undergrad/policy.html).

Project Objectives

The final project should be a synthesis of what you have learned over the course of the semester using OOP and Processing to produce a game. You should display a solid grasp on the programming constructs we have covered, and build something that you are excited about. This is probably the first time in the course that you can do whatever you would like to do, instead of doing what you were told to do. This project will give you practice in working with OOP design and implantation, designing Graphical User Interfaces (GUI), and many other concepts we learned this semester. Additionally, it is a good exercise in decomposing larger problems into smaller and more manageable parts.

A word of advice: choose a game that you enjoy playing yourself as you will be "engaged" to it for a few weeks. Also, it should be a game you are capable of developing yourself and would allow you to improve your programming skills whilst being implemented. The game should neither be trivial nor very difficult. A good example of a game idea is the popular Snake Game (You can make modifications): https://iq.opengenus.org/snake-game-java/
Or similar to the complete game that will be shown in class using processing.

The projects should be pursued in groups of three (same group as you have now) and you should clearly mention the group members and the group name (i.e., team 1) in the project proposal. You are not allowed to switch partners after this week! If you are having problems with your group,

please report these problems asap to us as this is very important to the success of your group. If you wait to last minute, then we will not be able to help you and your grade will be affected.

Game Requirements:

- Use Processing/Java
- Keyboard and/or mouse interaction (events)
- Object-oriented programming to create appropriate Classes/objects for the game (minimum of 3 classes which includes the main class (file) with setup() and draw()). Use as many features of OOP such as inheritance (One class inheritance from a super class) and composition (object contains another object).
- A score/result indicating the performance of the player or a win/lose indication
- A complete level or a set of challenges that the player can play
- Use images
- Sound (optional, extra credit)

The project consists of the following (there should be ONLY one submission per group by the team leader):

1) Project Proposal due on Saturday 12/7th by 11:55pm via Bright Space (*You will get feedback/comments by the 8th from the graders via* Bright Space):

- A project proposal of a game of your choice in written form (as a pdf and a max of 2 pages).

- Provide a name for your game and indicate the names and emails of all of your team members.

- The proposal should **provide a brief description (one or two sentences) of your idea (Game). Include a list of features you plan to integrate into the game**. This is to provide an idea of the game (It doesn't have to be very detailed or perfect). The list of features and strategy of the game (win, or lose) and you should also include responsibilities of each group member and the deadlines as well for each task.

- To support your idea, you can include a **mock design/screenshot/drawing** of your proposed game.

- You should also provide a UML diagram for each class you plan to integrate in the game. You can make changes to classes later on. The UML for all classes can be included in this proposal or in a separate pdf (outside of the two-page proposal required from above).

- You should be planning on meeting at least 4 times per week for a few hours. Make sure that you are integrating the game with each change (daily). Make copies of the game as mistakes happen.

- The project proposal must be submitted via **Bright Space** (see due date below). You will receive feedback on your proposal by the evening of the next day.

2) **Team weekly report due on Saturday 12/12th by 11:55pm via Bright Space:**

   - The design of the game, strategy and at least two classes should be done. You should have 70% of the functionality of the game done and also you should have a working game (simple play mode of the game). Zip all the files and submit via Bright space along with the .pdf report from below.

   - **Weekly report as a .PDF**: The weekly report (a .pdf) should Include previous information provided in the project proposal again such as the names and emails of all of your team members, the name of the project, brief description of the game and strategies (win and lose), scores, previous responsibilities and future responsibility/contribution of each member, the UML of all of your classes.

   - You should be planning on meeting at least 4 times per week for a few hours. Make sure that you are integrating the game with each change (daily). Make copies of the game as mistakes happen.

   - The project proposal must be submitted via **Bright Space** (see due date below). You will receive feedback on your proposal by the evening of the next day.

3) **Project Final Submission due via BrightSpace (one submission by the team leader) on December 17, by midnight):**
   - **Game Requirements:**
     - Use Processing/Java
     - Keyboard and/or mouse interaction (events)
     - Object-oriented programming to create appropriate Classes/objects for the game (minimum of 3 classes)
     - A score/result indicating the performance of the player or a win/lose indication
     - A complete level or a set of challenges that the player can play
     - **Please note that there will be no extension!**

Submission

The submission should include your presentation and all files required to run your project and/or reproduce your results. The code itself should also be clearly documented. In addition, a sample screenshot should be submitted that adequately represents what the project is about. Also, one submission on **Bright Space** per group is sufficient (By the team leader).

In summary, we look at:

- Code structure, style and documentation
- Proper use of OOP design and implementation for classes, and objects.
- No sudden crashes or apparent bugs

- Overall game design and graphics
- Complexity of the project (strategy of the game)
- Clean design for the Game (Graphic user Interface).
- Submit a final weekly report with team member reconstitutes, extra credit, UML and final description for the game and strategy used and scores.

Please submit a zip file containing **all** files of your assignment to **Bright Space**. Submissions via email are not accepted. Late submissions will NOT be accepted.

Note that your solution must work using Pressing/Java. In case your code does not work, your submission will not be graded.

Grading

Extra credit for added features!

Criterion	Points (Total: 15)
Project proposal submission	2
Weekly reports: one due on the 12th and one is due during the final submission by the 17th. The weekly report should be a .pdf and should include responsibilities(contributions) of members, UML, along with all the completed game files.)	2
Program clarity, comments, style and elegance of solution	1
Final Submission (complete OOP design: classes and objects, game interaction, game design, strategy and technical features, scores, level of complexity of the game, creativity and innovation)	10

Extra Credit: There will be up to 5 extra credits for added features such as providing multiple levels, providing more than 3 classes, adding sound, and so on. Make sure to include all of the extra credit features in the final weekly report when.